Implementation of Different Low Power Multipliers Using Verilog

Koteswara Rao Ponnuru
M. Tech, Assistant Professor
SRK Institute of Technology
Vijayawada, A.P., India

Shabeena Begum Mohammad
M. Tech, Assistant Professor
SRK Institute of Technology
Vijayawada, A.P., India

Abstract— **Low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance systems. Optimizing the speed and area of the multiplier is a major design issue. Multiplication represents a fundamental building block in all DSP tasks. The objective of a good multiplier is to provide a physically compact, good speed and low power consumption. To save significant power consumption of a VLSI design it is a good direction to reduce its dynamic power that is the major part of total power consumption. Two methods are common in current implementations: regular arrays and Wallace trees. The gate-level analyses have suggested that not only are Wallace trees faster than array schemes, they also consume much less power. However these analyses did not take wiring into account, resulting in optimistic timing and power estimates. Continuous advances of microelectronic technologies make better use of energy, encode data more effectively, reduce power consumption, etc. Particularly, many of these technologies address low-power consumption to meet the requirements of various portable applications. In these application systems, a multiplier is a fundamental arithmetic unit and widely used in circuits. I compare results for 8bit-width the working of different multipliers by comparing the power consumption by each of them. The result of my paper helps us to choose a better option between serial and parallel multiplier in fabricating different systems. Multipliers form one of the most important components of many systems. So, by analyzing the working of different multipliers helps to frame a better system with less power consumption and lesser area.**

Keywords: Array Multiplier; Bypassing Technique; Parallel Multiplier; Serial Multiplier; Wallace tree multiplier.

I. INTRODUCTION

There has been renewed interest in basic digital arithmetic in the last few years, driven originally by the migration of fast floating point hardware into standard microprocessors, and more recently by the demand for multimedia functionality in both desktop and portable systems. In the latter case, the demands of DSP style systems for both high throughput (e.g., for voice, video) and low energy consumption has spawned new work in low power circuit styles, DSP synthesis, and examination of basic tradeoffs in different arithmetic styles. Our interest is in the basic building blocks of arithmetic circuits, in particular, short word width (8-bit) multipliers of the type that dominate in DSP applications.

For individual arithmetic blocks, different gate-level architectures have a substantial impact on hardware size, layout complexity, speed and power. For basic adder types (ripple, look ahead, skip, bypass, etc.), the power tradeoffs were clearly mapped by Callaway and Swartz Lander. For example, ripple adders are slowest but use the least energy, whereas speculative styles such as bypass adders are fast but consume much more power, since they perform computations they later discard.

The tradeoffs for basic multipliers, however, are much less well characterized. Alter native designs focus on the manner in which the trapezoidal partial product arrays of individual bit-wise products are reduced (summed) to produce the final product. Array styles use a regular 2-D grid of adders for this reduction. Compact and easy to lay out, the arrays perform this reduction in gate depth that is linear in the bit width. At the other end of the spectrum, Wallace tree styles use a log-depth tree network for this reduction. Faster, but irregular, they trade ease of layout for speed. Although the speed-size tradeoffs for these two styles are fairly well characterized, the power tradeoffs are not well understood.

For example, Bellaouar and Elmasry suggest that Wallace tree styles are best avoided for low power applications, since the excess wiring is likely to consume extra power. On the hand, Callaway and Swartzlander demonstrate quantitatively that switching activity within just the partial product reduction hardware is substantially better for the tree over the array if one ignores the wires completely.

II. TECHNIQUES FOR IMPLEMENTING MULTIPLIERS

A. Serial Multiplier

This is the very first digit-serial systolic array for modular multiplication. The proposed architecture is highly regular and modular and thus well suited to VLSI implementation. The important feature of the proposed architecture is that different throughput performances can be easily achieved simply by varying the digit size. If the digit size is chosen appropriately the goal of this paper was to design, build, simulate and fabricate a 4-bit And 8-bit serial Multiplier. The reason I chose the serial multiplier is because it was the best circuit that tested all the crucial skills necessary for a successful circuit designer while at the same time being within the scope of this paper.

In building these components, integrating them together, and coordinating the timing and control of these elements, we were able to deal with many of the challenges an IC designer faces on a daily basis. Serial multiplication of two 4-bit numbers is hence the addition of four binary numbers, rows 1, 2, 3, and 4 in the above example. Row 1 is the multiplier times the 1st bit of the multiplicand, bit 1. Row 2 is the result of the multiplier times the 2nd bit of the multiplicand, bit 1, shifted to left by one 0. The process continues for each of the four rows, however, at each row the result is shifted left by an additional 0.It is important to note that at each row, the least significant bit (LSB) corresponds to an actual bit in the product.

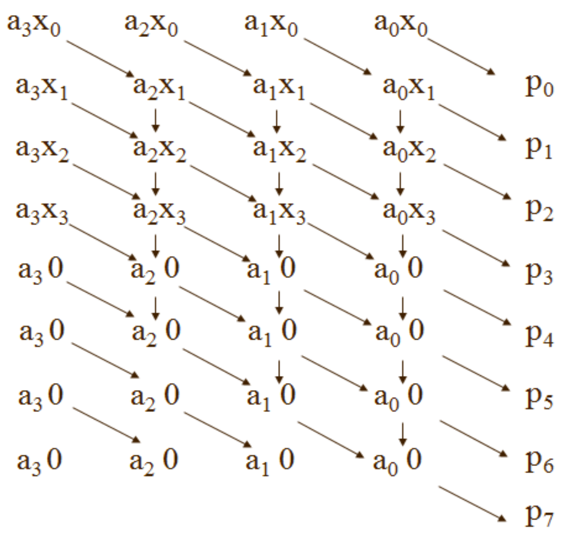
a_3x_0 a_2x_0 a_1x_0 a_0x_0
a_3x_1 a_2x_1 a_1x_1 a_0x_1
a_3x_2 a_2x_2 a_1x_2 a_0x_2
a_3x_3 a_2x_3 a_1x_3 a_0x_3
$a_3\,0$ $a_2\,0$ $a_1\,0$ $a_0\,0$
p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7

Figure 1. structure for serial multiplier

This bit can be saved as part of the product. The 3 most significant bits at each row are then summed with the next row's bits to produce a partial sum. This process continues until all four rows have been completed. At this point four least significant bits have been determined and the remaining four bits make up the four most significant bits of the product. This fact was critical in designing the serial multiplier.

Bit-serial computations have been studied extensively during the last decade. Complete VLSI design methodologies based on bit-serial arithmetic have been built, particularly for the VLSI implementation of digital signal processing (DSP) algorithms. However, the disadvantage of the bit-serial systems is that they are considered to be very slow for many applications such as radar, sonar, and video and image processing. For higher throughput DSP applications, however, it is clear that a move towards higher bit-width operations is necessary, where the solution often used is bit-parallel computation. However, the bit-parallel systems require a very large amount of silicon area, communication overhead, and pin-out.

The concept of digit serial implementations have been proposed in recent years Several approaches have been proposed to design digit serial-parallel architectures based on two's complement number representation. Early on most of the proposed design methodologies use the bit-level cellular arrays as their starting point. The major drawback of the architectures based on these approaches is that they cannot be pipelined at a sub-digit level due to the existence of carry feedback loops. This has severely limited their throughput, which could be a major obstacle for high speed applications. In digit serial-parallel systems, the throughput rate often represents the overriding factor dictating system performance.

1) Design of the digital Serial Multiplier

The radix-2n algorithm can be considered as a generalization of the binary arithmetic where the simplest building block is an AND gated Full Adder. Using the radix-2n arithmetic and assuming unsigned numbers, two N-bit numbers, X and Y, can be divided into *K* digits of n-bit each and can be written as

$$X = \sum_{i=0}^{K-1} X_i 2^{in} \qquad Y = \sum_{j=0}^{K-1} Y_j 2^{jn} \tag{1}$$

Where Xi represents the i th digit of the X and Yj is the j th digit of Y. The product, P, of two N bit Numbers, X and Y, can be written using the radix-2n arithmetic as

$$P = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} X_i Y_j 2^{(i+j)n} \tag{2}$$

Equation 2 can be computed using the dependency graph for K=4 where each node performs the multiplication of two n-bit digits, Xi and Yj, and adds the product to two other n-bit digits, Sin and Cin. The result will always be a 2-digit number, the most significant digit (MSD) is the carry digit, Cout, and the least significant digit (LSD) is the sum digit, Sout.

In serial-serial computation, at any one cycle the data is entered as a pair of digits, (X_K, Y_K), having the same significance, *k*. The products, X_KY_K, are computed at the nodes on the diagonal line i=j. In the index space (i, j), (i+j) represents a family of lines perpendicular to the line i=j and determines the significance. In serial-serial computation, the family of lines k=i+j should also define time.

The reason is that, at every cycle a new pair of digits (X_K, Y_K) enters the computation along one of the lines (i+j). As a result, no single index is used alone to define either time or space. Both Indices i and j are used to define time and hence a single projection along either i or j index will not result in digit serial-serial architecture.

B. Array Multiplier

To achieve high execution speed, parallel array multipliers are widely used. These multipliers tend to consume most of the power in DSP computations, and thus power efficient multipliers are very important for the design of low power DSP systems. A new method is proposed for array multipliers that can reduce power and area of the multiplier. The array multiplier follows the Carry Save method.

The advantage of the array multiplier is its regular structure, which leads to a dense layout, ideal for fabrication. In array multipliers, the counters and compressors are connected in a serial fashion for all bit slices of the Partial Product parallelogram. As can be seen in Figure the array topology is a two-dimensional structure that fits nicely on the VLSI planar process Multiplication is an essential arithmetic operation for common DSP (Digital Signal Processors) and Microprocessor Applications.

In recent years the researchers are emphasizes on three areas i.e. power, speed and area. Need of more application son a single processor will increase the number of transistors on a chip and cause increases to power consumption. Hence among the three fields one of the most important areas to be concentrates the power.

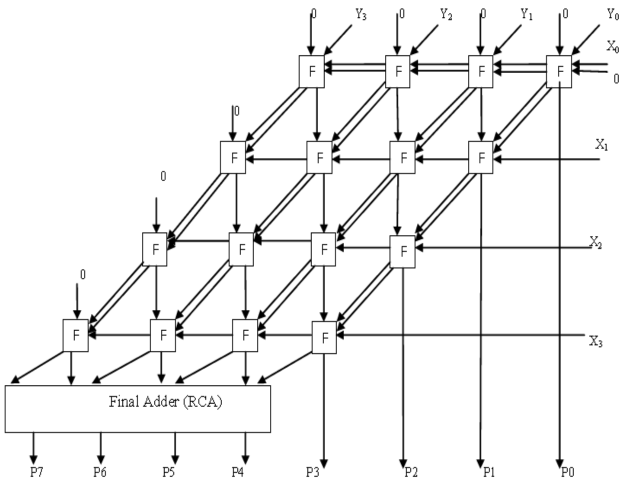

Figure 2. Array structured multiplier

In the Carry Save Addition method, the first row can be designed with either Half-Adders or Full-Adders. We have to multiply to bits (one partial product) each from X and Y. If the first row of the partial products is implemented with full adders, then the third input i.e. Cin will be considered as 0. The carries of each full adder can be diagonally forwarded to the next row of the adder. The resulting multiplier is said to be Carry Save Multiplier, because the carry bits are not immediately added, but rather are saved for the next stage.

The basic idea is to implement the design with full adders only. Hence in the design if the full adders have two input data at any stage, the third input is considered as zero. In the final stage, carries and sums are merged in a carry-propagate (e.g. ripple carry or carry-look ahead) adder stage.

C. *Braun Multiplier*

It is a simple parallel multiplier generally called as carry save array multiplier. It has been restricted to perform signed bits. The structure consists of array of AND gates and adders arranged in the iterative manner and no need of logic registers. This can be called as non – addictive multipliers.

The internal structure of the full adder can be realized using FPGA. Each products can be generated in parallel with the AND gates. Each partial product can be added with the sum of partial product which has previously produced by using the row of adders. The carry out will be shifted one bit to the left or right and then it will be added to the sum which is generated by the first adder and the newly generated partial product.

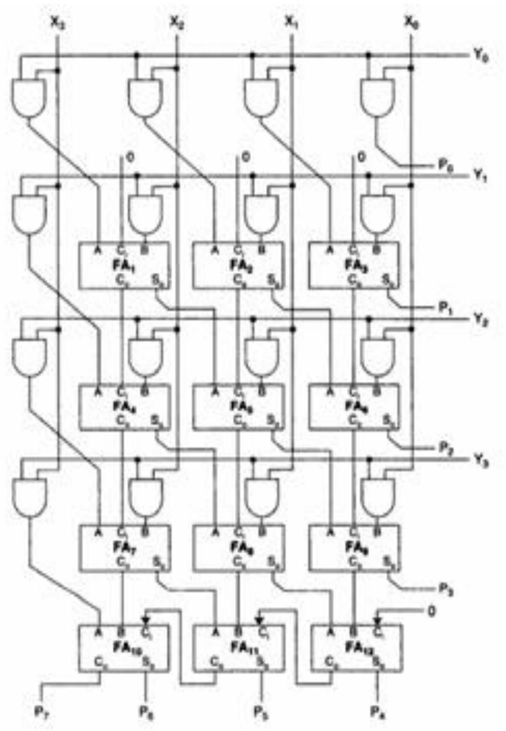

Figure 3. Block diagram of Braun Multiplier

The shifting would carry out with the help of Carry Save Adder (CSA) and the Ripple carry adder should be used for the final stage of the output. Braun multiplier performs well for the unsigned operands that are less than 16 bits in terms of speed, power and area. But it is simple structure when compared to the other multipliers. Fig3 shows the block diagram of Braun multiplier.

D. *Carry Bypass Multiplier*

The carry bypass multiplier is considerably faster than array multiplier but slower than Wallace tree multiplier the complexity of the Carry bypass/skip Multiplier is low compared Wallace tree and but high in case of ripple Carry Multiplier because it consist of an n bit CBA/CSKA (carry bypass or skip adder) is made up of n full adder gates which are grouped together into blocks whose size(i.e. the number of full adder per blocks) has to be properly chosen to minimize the time needed for a computation .

The CBA architecture can be derived from a simple ripple carry adder (obtained cascading n full adders) by stating that ,when some contiguous full adders work in propagate (i.e ,each of them has a carry output equal to the carry input),they can be bypassed to evaluate the carry output of the last one ,since it is equal to the carry input of the first.

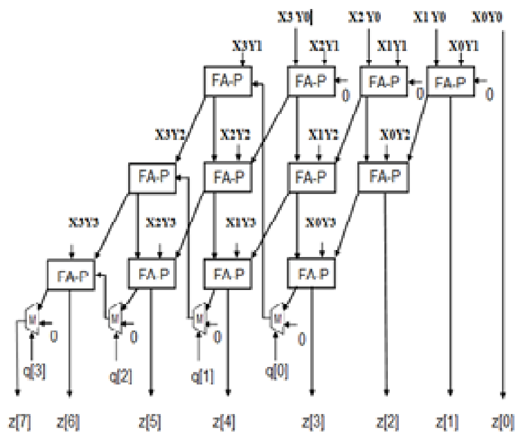

Figure 4. Carry Bypass Multiplier

Hence ,in a CBA the full adders are divided in to groups ,each of them is bypassed by a multiplexer if its full adders are all in propagate .the resulting architecture of a multiplier with carry bypass adders is shown in Fig4.

1) Bypassing Technique

Dynamic power consumption can be reduced by bypassing method when the multiplier of the full adders with the propagation (FA-P) cells of the propagation product is high/logic 1 input otherwise the output carry of the carry bypass adder output carry (Cout) come from critical path. Multiplexer propagates the carry input to the carry output. When BP = 1, the input carry (Cin) output of the full adder is passed.

Fig. 5 shows the easy path for carry bypass multiplier.

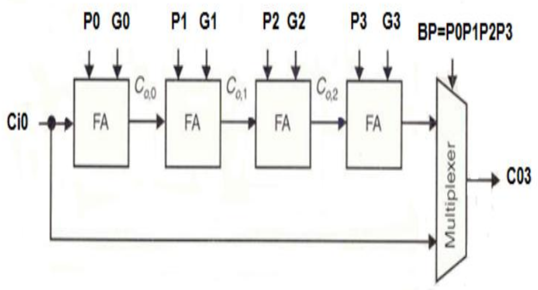

Figure 5. Simple path in CBA cell

Fig. 6 shows the critical path for carry bypass multiplier.

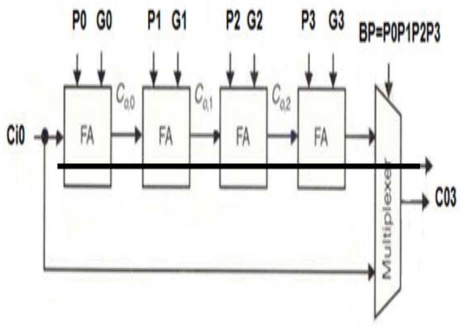

Figure 6. Critical path in CBA cell

To skip the carry from the ripple carry adder MUX is used gates can be used, as ideal switches with small power consumption, propagation delay and small area. To study the proposed design we have consider column bypassing multiplier in which columns of adders input carry’s are bypassed. In this multiplier, the operations in a column input carry can be bypassed if the corresponding propagation product (BP) bit is 1. The advantage of this multiplier is it eliminates the extra correcting circuit. The column bypassing multiplier (CBM) only needs one multiplexer in a adder cell. When y (BP) is 0 then the corresponding column cells are functioning necessarily.

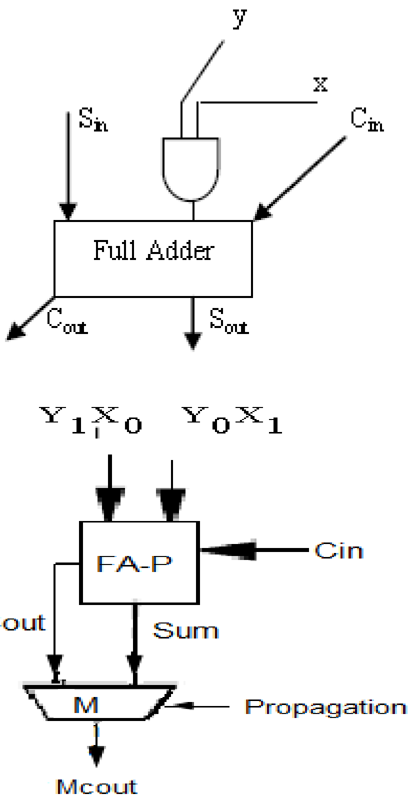

Figure 7. (a) Full adder cell (b) Full adder bypass

In all these cells the partial products xi × yj and the carry inputs are zero for i = 0, 1… n-1.and this chain does not contribute to the formation of the product. To achieve all of the above we can replace the Full Adder cell shown in Fig 7(a) with the cell in Fig 7(b) called the Full Adder with propagation (FA_P) cell.

Adder which is used to add carries and sums of the multiplier in the conventional is removed in this method. Here the carries of the multiplier are not neglected. The carry of the fourth column of the 4x4 multiplier is given to the input of the fifth column instead of zero. The full adder at the top of the fifth column have only two input data, so the third is considered as zero in conventional multiplier. But in the proposed the carry of the fourth column is given to the input of the fifth column first adder. The use of a full adder is to add given inputs. The full adder of Ripple Carry Adder (RCA) can do the same functionality at the final addition stage. That why the carry of the fourth column is fed to the input of the first adder in the fifth column. In that adder the carry merges with the two inputs.

Then the carry of the fifth column is forwarded to the input of first adder of the sixth column so on. In this multiplier the carry of the seventh column of the adder is not neglected, it is considered as Most Significant Bit (MSB) of the multiplier. Due to elimination of four full adders at the final addition stage power and area can be trade off in the proposed design. The blocks are connected by 2:1 multiplexers, which can be placed into one or more level structures. The strategies proposed until now to find the optimum distribution of full adders into blocks are based on iterative algorithms, complex pencil-and-paper procedures, or approximate closed-form solutions. However, design strategies proposed do not provide an insight on the optimum block size, are based on unrealistic hypothesis or are tedious to be applied. In this paper a novel optimization strategy of block sizes for one-level Carry Bypass Adders is proposed.

It is based on a first analytical sizing of an optimum but incomplete adder, and successively missing bits are properly added to achieve the desired number of bits with minimum delay increase. The strategy proposed is systematic and general, independent of the technology used, and it is suitable for pencil-and paper design. The validity of the proposed strategy is checked by comparing the resulting delay to optimum results in for different numbers of bits, full adder and multiplexer delays.

E. Wallace Tree Multiplier

The Wallace tree multiplier is considerably faster than a simple array multiplier because its height is logarithmic in word size, not linear. However, in addition to the large number of adders required, the Wallace tree's wiring is much less regular and more complicated. As a result, Wallace trees are often avoided by designers, while design complexity is a concern to them.

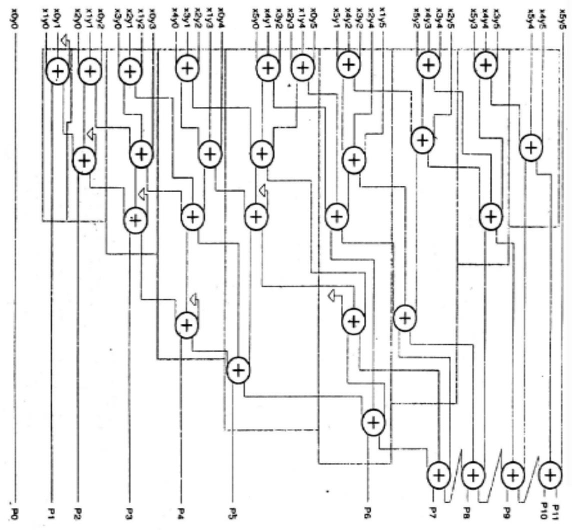

Figure 8. Wallace tree block diagram

Wallace tree styles use a log-depth tree network for reduction. Faster, but irregular, they trade ease of layout for speed.

Wallace tree styles are generally avoided for low power applications, since excess of wiring is likely to consume extra power. While subsequently faster than Carry-save structure for large bit multipliers, the Wallace tree multiplier has the disadvantage of being very irregular, which complicates the task of coming with an efficient layout. The Wallace tree multiplier is a high speed multiplier. The summing of the partial product bits in parallel using a tree of carry-save adders became generally known as the "**Wallace Tree".** Three step processes are used to multiply two numbers.

- Formation of bit products.
- Reduction of the bit product matrix into a two row matrix by means of a carry save adder.
- Summation of remaining two rows using a faster Carry Look Ahead Adder (CLA).

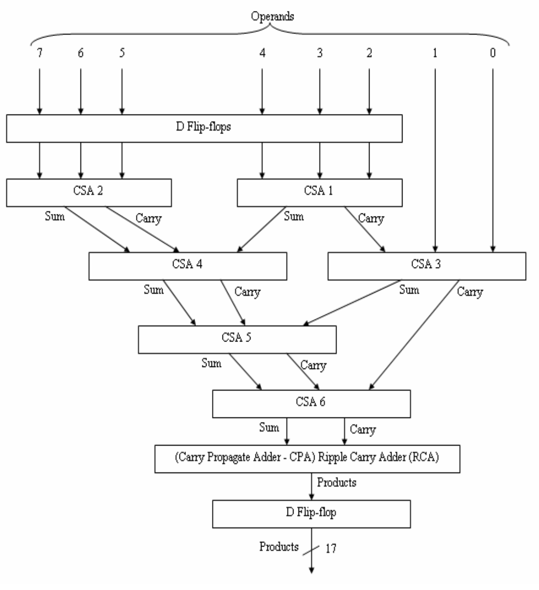

Figure 9. Wallace Tree Block Diagram

In order to design an **n-bit Wallace tree Multiplier** (Generic: =N) an algorithm was derived from the flow diagram developed above. The flow diagram above shows the intermediate state reductions of the multipliers are being done by Carry save adders and half adders while the final step additions being done by a Carry Look Ahead Adder.

III. SIMULATION RESULTS

In this paper, the performance of the high speed low power multipliers like Serial, Array, parallel (Braun), Wallace and carry bypass/skip multipliers have been evaluated. These multipliers can be implemented using Verilog HDL coding. In order to get the power report and delay report, I am synthesizing these multipliers using Xilinx 9.2i.

The result of my paper helps us to choose a better option between serial and parallel multiplier in fabricating different systems. Multipliers form one of the most important components of many systems. So by analyzing the working of different multipliers helps to frame a better system with less power consumption and lesser area.

A. *Serial Multiplier*

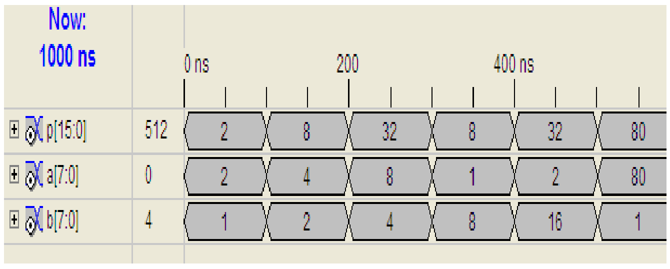

Figure 10. Simulation results for serial multiplier

B. *Array Multiplier*

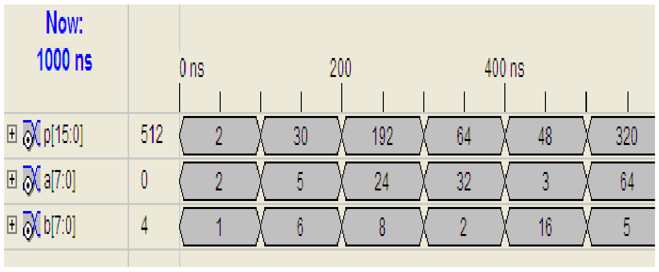

Figure 11. Simulation results for array multiplier

C. *Braun Multiplier*

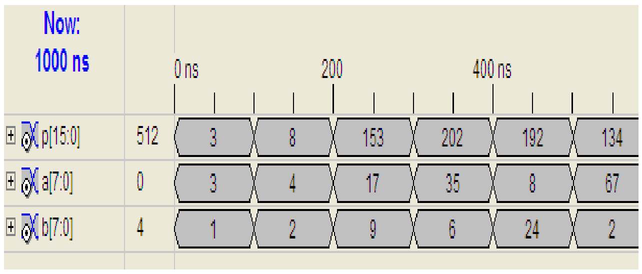

Figure 12. Simulation results for Braun multiplier

D. *Wallace Tree Multiplier*

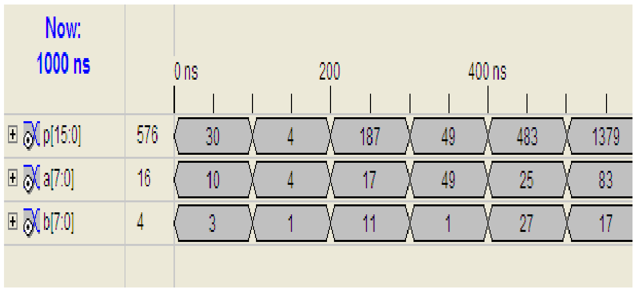

Figure 13. Simulation results for Wallace tree multiplier

E. *Carry Bypass Multiplier*

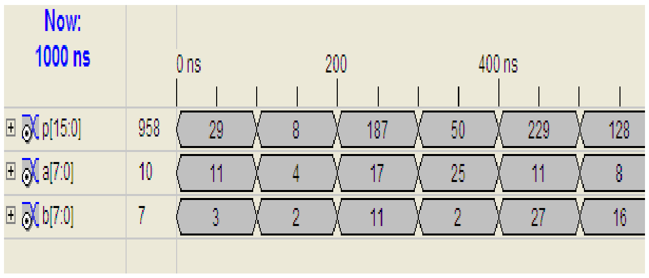

Figure 14. Simulation results for Carry bypass Multiplier

TABLE I. COMPARISON

S. No	**Name of the multiplier**	**Power Consumed**	**Delay**
1.	Array multiplier	102mW	49.718ns
2.	Braun multiplier	127mW	27.841ns
3.	Serial multiplier	27mW	25.713ns
4.	Wallace tree multiplier	24mW	27.701ns
5.	Carry bypass multiplier	21mW	80.065ns

IV. CONCLUSION

In this paper, I implemented different types of multipliers like parallel, serial, Wallace, carry bypass and array multiplier using verilog with target device Spartan3 with speed grade of -5. I am calculating the power and delay of all the above multipliers with input frequency 2MHz and output capacitance of 1000Ff.The calculating power of the carry bypass multiplier is less comparing to other multipliers. So this multiplier can be used in low power designed devices.

REFERENCES

[1] Young-Ho Seo and Dong-Wook Kim,"*A new VLSI architecture of parallel multiplier-accumulator based on radix2 modified booth algorithm*," in *Proc.* IEEE transactions on very large scale integration (VLSI) systems, vol. 18, no. 2, pp.201-208, February 2010.

[2] C.N.Marimuthu and P.Thangaraj, "*low power high performance multiplier*," *Proc. ICGST-PDCS*, vol. 8, pp.31–38, dec 2008.

[3] K.H.Chen, K.C.Chao, J.I.Guo, J.S.Wang and Y.S. Chu, "*An efficient spurious power suppression technique (SPST) and its applications on MPEG-4 AVC/H.264 transform coding design*, "in Proc.IEEE Int. Symps. Low Power Electron. Des. 2005, pp.155–160.

[4] Z. Huang and M. D.Ercegovac, "*High-performance low-power left-to right array multiplier design*," IEEE Trans.Comput.,vol.54,no.3, pp.272–283,Mar.2005.

[5] H. Lee, "*A power-aware scalable pipelined Booth multiplier*," in Proc. IEEE Int.SOC Conf., 2004, pp.123–126.

[6] F. Elguibaly, "A fast parallel multiplier–accumulator using the modified Booth algorithm," *IEEE Trans. Circuits Syst.*, vol. 27, no. 9, pp.902–908, Sep. 2000.

[7] A. R. Cooper, "*Parallel architecture modified Booth multiplier*," *Proc.Inst. Electr. Eng. G*, vol. 135, pp. 125–128, 1988